

**REMARKS**

This communication is a full and timely response to the final Office Action dated October 10, 2011. Claims 1-9 and 12-15 remain pending, of which claims 1, 8, 9, 12-15 are the independent claims. By this communication, claims 1-3, 8, 9 and 12-15 are amended. Support for the amended subject matter can be found, for example, in FIGS. 3A and 3B and page 13 of the original application. Favorable reconsideration of the present application is respectfully requested in view of the foregoing Amendments and the following Remarks.

**Claim Rejections Under 35 U.S.C. § 103**

Claims 1-9 and 12-15 are rejected under 35 U.S.C. § 103(a) for alleged unpatentability over U.S. Patent No. 6,832,373 to O'Neill ("O'Neill") in view of U.S. Patent No. 7,657,886 to Chen et al. ("Chen"). The Applicants respectfully traverse this rejection.

Claim 1, as amended, recites, among other things, a method for in-place updating comprising comparing a form of a received update package, which indicates a direction in which said received update package updates a file, with a form of an old version to determine if the update package corresponds to said form of said old version.

Claim 1 further recites updating blocks in said old version in the direction of the received update package, if said form of said received update package corresponds to said determined form of said old version.

In the context of claim 1, a form of a file version is determined that indicates at which end of the file free space is available. Specifically, free space in the file can either be at the beginning of the file (e.g., at one end) or at the end of the file (e.g.,

the other end). When receiving an update package, the form of the update package (e.g., the direction in which the update package updates a file) is compared with the form of the old file version in order to determine if the update package can be used to update the file. For example, if a file version includes free space at the beginning of the file, an update package configured to update a file from the beginning of the file toward the end of the would be needed. Thus, according to claim 1, if the form of the update file, which indicates the direction in which it updates a file, corresponds to the form of the file (the order in which the file is to be updated), then updating can commence. In other words, in the context of claim 1, and in certain exemplary embodiments, an old version of a file is updated when a received update package is configured to update the file in direction (e.g., beginning to end) that is compatible with the form (e.g., free space at beginning) of the old file version, thereby updating the old file with one write per every block.

Thus, by performing computations and/or recording information pertaining to the direction necessary for update prior to the transmission of the update package, an update can be carried out (with one write per every block) without the use of auxiliary backup blocks, thus effectively reducing the time required for updating the old version on the device.

The Applicants respectfully submit that neither *O'Neil* nor *Chen*, whether considered alone or in proper combination, discloses or suggests the method with combination of steps and features, as recited in claim 1.

*O'Neil*, in contrast with claim 1, does not identify a form of an update package let alone identifying a direction in which the update package is configured to update a file. Moreover, *O'Neill* does not disclose or remotely suggest a form of an old version that indicates an end of the old version at which free space is available, and

further does not generate a new version wherein free space is an end opposite that of the old version.

In *O'Neill*, there are two methods by which a client device can receive and update software components. In a first method, a client device can send a request to an update device server including information such as type, model, and make of the device, desired update version, and/or code version currently being used by the device. In response, the server will either (1) send the appropriate version update package (by way of generating it or locating an archived version) to the client for installation or (2) send a response indicating the size of the file in order for the client device to determine if enough space exists in memory of the device to install the update. See, e.g., col. 11, ll. 32-45, col. 13, ll. 41-60. Thus, according to this method, upon sending a request, the client device (if enough space exists in memory) receives the appropriate update package and installs it.

In a second method, the update server will provide a list to a client device of all archived update packages that the client can choose from. The client may then select a desired update package, submit a request for the desired update package, and receive the desired update package for installation.

Neither of the above-described methods disclosed by *O'Neill* involves indicating which end of an old version file has available space. Nor does *O'Neill* disclose that a form of the received update package (at the client device) indicates a direction in which the update package updates a file. Because neither of these features is disclosed or remotely contemplated by *O'Neill*, a comparison of these two features cannot be and is not made.

Moreover, the space to which *O'Neill* refers, and the Office relies (see, Office Action, p. 7) is merely the available storage space in the memory of the device (i.e.,

RAM), and the allocation of space pertains to the client device writing current files stored in one area (e.g., RAM) to a second data area (e.g., onboard flash memory). In other words, files are moved around in order to free up space in the device's memory for the download transfer of an update package. *O'Neill* does not disclose that the free space is located at an end of an old version in memory, as recited in claim 1.

Regardless if *O'Neill* discloses methods of updating an old version via a received update package, the update package (namely a form of the update package pertaining to a direction in which the update package updates a version) is not compared with a form of an old version (namely an end of the version file where free space is available) upon a device receiving the update package. In contrast, in *O'Neill* merely discloses receiving update package, storing it in memory, and installing it.

While the Office appears to rely upon *O'Neill* for its disclosure pertaining to available space in memory, the Office also acknowledges that *O'Neill* fails to disclose or suggest that the alleged available space is located at an end of the old version file. In an effort to remedy this deficiency, the Office turns to *Chen*.

Regardless if *Chen* discloses block layout modifications upon updates, the layout modifications do not occur in a manner equivalent to the changes in free space (of old and new versions), of claim 1. Nor is free space within memory of *Chen* (at an end of a block layout) compared to a form of an update package that indicates *a direction in which a file can be updated*, as recited in claim 1. As disclosed in *Chen*, an update process can use a free/available block as a first block to be written with updated content. As an example provided by *Chen*, an initial layout may include a free block located at a beginning, and upon conclusion of the

update process, the free block may move to the end. See, e.g., col. 8, ll. 5-18.

However, in *Chen*, a free block may occur at any location. In particular, once a free block is located, the update process begins and uses that free block in order to write updated content. *Chen* does not disclose or remotely contemplate a comparison between the form of an old version, indicating an end of an old version where free space is located, and a form of a received update package. More specifically, *Chen* is entirely devoid of disclosure pertaining to forms of update packages *that indicate a direction in which the update packages can update version files*. Because *Chen* does not contemplate directions in which update packages can update a version file, a comparison between an update package and its associated direction for updating a file cannot be made with a form of a version file, which is in contrast with claim 1. Thus, *Chen* does not reconcile the deficiencies of *O'Neill* for failing to disclose or suggest the above-noted claim features.

In addition to the above-noted deficiencies, the Applicants respectfully submit that in *Chen*, the use of a memory management unit (MMU) is *required* in order to conduct efficient fault tolerant updates.

When an MMU is employed in *Chen*, the MMU is the component that enables the update to be more efficient (e.g., one write per updated block). In particular, the MMU maintains a mapping between the virtual memory and its physical address in non-volatile memory and manages the free space. In other words, the MMU indicates where free space is located (e.g., any location). When a free space is located, the updated content is written to the free memory. The original content, however, is maintained in its old logical block until the writing of the updated content to a different physical block is completed. This is repeated for the next block to be updated. Regardless if more efficient updating is accomplished by employing the

MMU, the MMU is *necessary* in *Chen* for the management of free space in memory, which is in contrast with the present claims.

The method of claim 1 improves upon the method of *Chen* by removing the MMU requirement. This is accomplished by (1) indicating, in the form of the update package, a direction in which the update package can update a version file and (2) comparing this form/direction with the form of the old version, which indicates at which end of the old version free space is available, as recited in claim 1. Such a comparison results in an appropriate/corresponding update package that can be used for the update, thereby resulting in efficient updating (one write per one block) without the need of an MMU to manage free space, as required in *Chen*.

When an MMU is not employed in *Chen*, an update requires two writes per block. In particular, a free memory block for backup purposes is first located, which can occur at any location, as discussed on page 12 of this paper. After a free block is located and written to with updated content, another free block is located. In the absence of an MMU, which points to free blocks, the update continues from the previously located free block in a given order, to locate the next free memory. Because this is an in-place update, the old block that was successfully updated to a free block is no longer required and then serves as free memory for the next phases (e.g., physically 'freeing' it). See, e.g., Col 8, ll. 11-16. *Chen*, however, is entirely silent as to (1) how a free block is moved without performing a write operation and (2) how an original block is marked as a free block (in the absence of a MMU) without performing a write operation. As a result, in the absence of a MMU in *Chen*, an update involve a *second* write for 'freeing' the old block.

Thus, the method of claim 1 achieves an efficient update with one write per updated block without the need for a free memory management unit (MMU), as in

*Chen*, by comparing the forms of both the update package (e.g., direction in which it updates) and the old version (e.g., at which end free space is available).

For at least the reasons set forth above, the Applicants respectfully submit that neither *O'Neill* nor *Chen*, whether considered alone or in proper combination, disclose or suggest, implicitly or explicitly, each and every claim feature of claim 1. Therefore, the combination of references cannot maintain a rejection of claim 1 under 35 U.S.C. 103. Accordingly, the Applicants respectfully submit that claim 1 is patentably distinct from *O'Neill* and *Chen* and request the rejection be withdrawn.

Claims 2-7 depend from independent claim 1 and therefore incorporate all of its subject matter. Accordingly, the Applicants respectfully submit that dependent claims 2-7 are patentably distinct from *O'Neill* and *Chen* for at least the reasons set forth above with respect to claim 1 and request the rejection be withdrawn.

Independent claims 8, 9 and 12-15, while different in scope, recite at least some of the same distinguishing features noted above with respect to claim 1. Therefore, arguments similar to those in connection with claim 1 are also applicable to claims 8, 9 and 12-15. Accordingly, the Applicants respectfully submit that independent claims 8, 9 and 12-14 are patentably distinct from *O'Neill* and *Chen* and request that the rejection be withdrawn.

**CONCLUSION**

In light of the foregoing, the Applicants respectfully request reconsideration and allowance of the above-captioned application. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is invited to telephone the undersigned at the number provided.

Respectfully submitted,

BUCHANAN INGERSOLL & ROONEY PC

Date: December 12, 2011

By:

A handwritten signature in black ink, appearing to read "Charles F. Wieland III", is written over a horizontal line.

Charles F. Wieland III  
Registration No. 33,096

**Customer No. 21839**  
703 836 6620